Update-Efficient Error-Correcting Regenerating Codes

Yunghsiang S. Han, *Fellow*, *IEEE*, Hong-Ta Pai, *Senior Member*. *IEEE*, Rong Zheng, *Senior Member*. *IEEE*, Pramod K. Varshney, *Life Fellow*, *IEEE*

Abstract

Regenerating codes provide an efficient way to recover data at failed nodes in distributed storage systems. It has been shown that regenerating codes can be designed to minimize the per-node storage (called MSR) or minimize the communication overhead for regeneration (called MBR). In this work, we propose new encoding schemes for $[n, d]$ error-correcting MSR and MBR codes that generalize our earlier work on error-correcting regenerating codes. We show that by choosing a suitable diagonal matrix, any generator matrix of the $[n, \alpha]$ Reed-Solomon (RS) code can be integrated into the encoding matrix. Hence, MSR codes with the least update complexity can be found. By using the coefficients of generator polynomials of $[n, k]$ and $[n, d]$ RS codes, we present a least-update-complexity encoding scheme for MBR codes. An efficient decoding scheme is proposed that utilizes the $[n, \alpha]$ RS code to perform data reconstruction for MSR codes. The proposed decoding scheme has better error correction capability and incurs the least number of node accesses when errors are present. A new decoding scheme is also proposed for MBR codes that can correct more error-patterns.

Parts of this work were submitted to the IEEE International Symposium on Information Theory (ISIT 2013). This work was supported in part by CASE: The Center for Advanced Systems and Engineering, a NYSTAR center for advanced technology at Syracuse University; the National Science of Council (NSC) of Taiwan under grants no. 99-2221-E-011-158-MY3 and NSC 101-2221-E-011-069-MY3; US National Science Foundation under grant no. CNS-1117560 and McMaster University new faculty startup fund. Han's work was completed during his visit of Syracuse University from 2012 to 2013.

Han is with the Dept. of Electrical Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan (e-mail: yshan@mail.ntust.edu.tw), Pai is with the Graduate Institute of communication Engineering, National Taipei University, Taiwan, R.O.C., Zheng is with the Dept. of Computing and Software, McMaster University, Hamilton, ON, Canada, and Varshney is with the Dept. of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY USA.

Index Terms

Distributed storage, Regenerating code, Reed-Solomon code, Decoding

I. Introduction

Cloud storage is gaining popularity as an alternative to enterprise storage where data is stored in virtualized pools of storage typically hosted by third-party data centers. Reliability is a key challenge in the design of distributed storage systems that provide cloud storage. Both crash-stop and Byzantine failures (as a result of software bugs and malicious attacks) are likely to be present during data retrieval. A crash-stop failure makes a storage node unresponsive to access requests. In contrast, a Byzantine failure responds to access requests with erroneous data. To achieve better reliability, one common approach is to replicate data files on multiple storage nodes in a network. There are two kinds of approaches: duplication (Google) [1] and erasure coding [2], [3]. Duplication makes an exact copy of each data and needs lots of storage space. The advantage of this approach is that only one storage node needs to be accessed to obtain the original data. In contrast, in the second approach, erasure coding is employed to encode the original data and then the encoded data is distributed to storage nodes. Typically, more than one storage nodes need to be accessed to recover the original data. One popular class of erasure codes is the maximum-distance-separable (MDS) codes. With $[n, k]$ MDS codes such as Reed-Solomon (RS) codes, k data items are encoded and then distributed to and stored at n storage nodes. A user or a data collector can retrieve the original data by accessing *any* k of the storage nodes, a process referred to as *data reconstruction*.

Any storage node can fail due to hardware or software damage. Data stored at the failed nodes need to be recovered (regenerated) to remain functional to perform data reconstruction. The process to recover the stored (encoded) data at a storage node is called *data regeneration*. A simple way for data regeneration is to first reconstruct the original data and then recover the data stored at the failed node. However, it is not efficient to retrieve the entire B symbols of the original file to recover a much smaller fraction of data stored at the failed node. *Regenerating codes*, first introduced in the pioneer works by Dimakis *et al.* in [4], [5], allow efficient data regeneration. To facilitate data regeneration, each storage node stores α symbols and a total of d surviving nodes are accessed to retrieve $\beta \leq \alpha$ symbols from each node. A trade-off exists between the storage overhead and the regeneration (repair) bandwidth needed for data regeneration. Minimum Storage Regenerating (MSR) codes first minimize the amount of data stored per node, and then the repair bandwidth, while Minimum Bandwidth Regenerating (MBR) codes carry out the minimization in the reverse order. There have been many works that focus on the design of regenerating codes [6]–[13]. Recently, Rashmi *et al.* proposed optimal exact-regenerating codes that recover the stored data at the failed node exactly (and thus the name exact-regenerating) [13]; however, the authors only consider crash-stop failures of storage nodes. Han *et al.* extended Rashmi's work to construct error-correcting regenerating codes for exact regeneration that can handle Byzantine failures [14]. In [14], the encoding and decoding algorithms for both MSR and MBR error-correcting codes were also provided. In [15], the code capability and resilience were discussed for error-correcting regenerating codes.

In addition to bandwidth efficiency and error correction capability, another desirable feature for regenerating codes is *update complexity* [16], defined as the maximum number of encoded symbols that must be updated while a single data symbol is modified. Low update complexity is desirable in scenarios where updates are frequent. Clearly, the update complexity of a regenerating code is determined by the number of non-zero elements in the row of the encoding matrix with the maximum Hamming weight. The smaller the number, the lower the update complexity is.

One drawback of the decoding algorithms for MSR codes given in [14] is that, when one or more storage nodes have erroneous data, the decoder needs to access extra data from many storage nodes (at least k more nodes) for data reconstruction. Furthermore, when one symbol in the original data is updated, all storage nodes need to update their respective data. Thus, the MSR and MBR codes in [14] have the maximum possible update complexity. Both deficiencies are addressed in this paper. First, we propose a general encoding scheme for MSR codes. As a special case, least-update-complexity codes are designed. We also design least-update-complexity encoding matrix for the MBR codes by using the coefficients of generator polynomials of the $[n, k]$ and $[n, d]$ RS codes. Second, a new decoding algorithm is presented for MSR codes. It not only provides better error correction capability but also incurs low communication overhead when errors occur in the accessed data. Third, we devise a more efficient decoding scheme for the MBR codes that can correct more error patterns compared to the one in [14].

The rest of this paper is organized as follows. Section II gives an overview of error-correcting regenerating codes. Section III presents the least-update-complexity encoding and efficient decoding schemes for error-correcting MSR regenerating codes. Section IV demonstrates the least-update-complexity encoding of MBR codes and the corresponding decoding scheme. Section V details evaluation results for the proposed decoding schemes. Section VI concludes the paper with a list of future work. Since only error-correcting regenerating codes are considered in this work, unless stated otherwise, we refer to error-correcting MSR and MBR codes as MSR and MBR codes in the rest of the paper.

II. Error-Correcting Regenerating Codes

In this section, we give a brief overview of data regenerating codes, the MSR and MBR code constructions in [14].

A. Regenerating Codes

Let α be the number of symbols stored at each storage node and $\beta \leq \alpha$ the number of symbols downloaded from each storage during regeneration. To repair the stored data at the failed node, a helper node accesses d surviving nodes. The design of regenerating codes ensures that the total regenerating bandwidth be much less than that of the original data, B. A regenerating code must be capable of reconstructing the original data symbols and regenerating coded data at a failed node. An $[n, k, d]$ regenerating code requires at least k nodes to ensure successful data reconstruction, and d surviving nodes to perform regeneration [13], where n is the number of storage nodes and $k \leq d \leq n - 1$.

The cut-set bound given in [5], [6] provides a constraint on the repair bandwidth. By this bound, any regenerating code must satisfy the following inequality:

$$B \leq \sum_{i=0}^{k-1} \min\{\alpha, (d-i)\beta\} \ . \tag{1}$$

From (1), α or β can be minimized achieving either the minimum storage requirement or the minimum repair bandwidth requirement, but not both. The two extreme points in (1) are referred to as the minimum storage regeneration (MSR) and minimum bandwidth regeneration (MBR) points, respectively. The values of α and β for the MSR point can be obtained by first minimizing

α and then minimizing β:

$$\begin{aligned} \alpha &= d-k+1 \\ B &= k(d-k+1) = k\alpha \ , \end{aligned} \tag{2}$$

where we normalize β and set it equal to 1.[1] Reversing the order of minimization we have α for MBR as

$$\begin{aligned} \alpha &= d \\ B &= kd - k(k-1)/2 \ , \end{aligned} \tag{3}$$

while $\beta = 1$.

There are two categories of approaches to regenerate data at a failed node. If the replacement data is exactly the same as that previously stored at the failed node, we call it *exact regeneration*. Otherwise, if the replacement data only guarantees the correctness of data reconstruction and regeneration properties, it is called *functional regeneration*. In practice, exact regeneration is more desirable since there is no need to inform each node in the network regarding the replacement. Furthermore, it is easy to keep the codes systematic via exact regeneration, where partial data can be retrieved without accessing all k nodes. The codes designed in [13], [14] allow exact regeneration.

B. MSR Regenerating Codes With Error Correction Capability

Next, we describe the MSR code construction given in [14]. Here, we assume $d = 2\alpha$. The information sequence $\boldsymbol{m} = [m_0, m_1, \ldots, m_{B-1}]$ can be arranged into an information vector $U = [Z_1 Z_2]$ with size $\alpha \times d$ such that Z_1 and Z_2 are symmetric matrices with dimension $\alpha \times \alpha$. An $[n, d = 2\alpha]$ RS code is adopted to construct the MSR code [14]. Let a be a generator of $GF(2^m)$. In the encoding of the MSR code, we have

$$U \cdot G = C, \tag{4}$$

[1]It has been proved that when designing $[n,k,d]$ MSR codes for $k/(n+1) \leq 1/2$. it suffices to consider those with $\beta = 1$ [13].

where

$$G = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ a^0 & a^1 & \cdots & a^{n-1} \\ (a^0)^2 & (a^1)^2 & \cdots & (a^{n-1})^2 \\ & & \vdots & \\ (a^0)^{d-1} & (a^1)^{d-1} & \cdots & (a^{n-1})^{d-1} \end{bmatrix},$$

and C is the codeword vector with dimension $(\alpha \times n)$.

It is possible to rewrite generator matrix G of the RS code as,

$$\begin{aligned} G &= \begin{bmatrix} 1 & 1 & \cdots & 1 \\ a^0 & a^1 & \cdots & a^{n-1} \\ (a^0)^2 & (a^1)^2 & \cdots & (a^{n-1})^2 \\ & & \vdots & \\ (a^0)^{\alpha-1} & (a^1)^{\alpha-1} & \cdots & (a^{n-1})^{\alpha-1} \\ (a^0)^{\alpha}1 & (a^1)^{\alpha}1 & \cdots & (a^{n-1})^{\alpha}1 \\ (a^0)^{\alpha}a^0 & (a^1)^{\alpha}a^1 & \cdots & (a^{n-1})^{\alpha}a^{n-1} \\ (a^0)^{\alpha}(a^0)^2 & (a^1)^{\alpha}(a^1)^2 & \cdots & (a^{n-1})^{\alpha}(a^{n-1})^2 \\ & & \vdots & \\ (a^0)^{\alpha}(a^0)^{\alpha-1} & (a^1)^{\alpha}(a^1)^{\alpha-1} & \cdots & (a^{n-1})^{\alpha}(a^{n-1})^{\alpha-1} \end{bmatrix} \quad (5) \\ &= \begin{bmatrix} \bar{G} \\ \bar{G}\Delta \end{bmatrix}, \quad (6) \end{aligned}$$

where $\bar{G}$ contains the first α rows in G, and Δ is a diagonal matrix with $(a^0)^\alpha,\ (a^1)^\alpha,\ (a^2)^\alpha, \ldots,\ (a^{n-1})^\alpha$ as diagonal elements, namely,

$$\Delta = \begin{bmatrix} (a^0)^\alpha & 0 & 0 & \cdots & 0 & 0 \\ 0 & (a^1)^\alpha & 0 & \cdots & 0 & 0 \\ & & \vdots & & & \\ 0 & 0 & 0 & \cdots & 0 & (a^{n-1})^\alpha \end{bmatrix}. \quad (7)$$

Note that if the RS code is over $GF(2^m)$ for $m \geq \lceil \log_2 n\alpha \rceil$, then it can be shown that $(a^0)^\alpha,\ (a^1)^\alpha,\ (a^2)^\alpha, \ldots,\ (a^{n-1})^\alpha$ are all distinct. According to the encoding procedure, the

α symbols stored in storage node i are given by,

$$U \cdot \begin{bmatrix} \boldsymbol{g}_i^T \\ (a^{i-1})^\alpha \boldsymbol{g}_i^T \end{bmatrix} = Z_1 \boldsymbol{g}_i^T + (a^{i-1})^\alpha Z_2 \boldsymbol{g}_i^T,$$

where $\boldsymbol{g}_i^T$ is the ith column in $\bar{G}$.

C. MBR Regenerating Codes With Error Correction Capability

In this section, we describe the MBR code constructed in [14]. Note that at the MBR point, $\alpha = d$. Let the information sequence $\boldsymbol{m} = [m_0, m_1, \ldots, m_{B-1}]$ be arranged into an information vector U with size $\alpha \times d$, where

$$U = \begin{bmatrix} A_1 & A_2^T \\ A_2 & \mathbf{0} \end{bmatrix}, \tag{8}$$

A_1 is a $k \times k$ symmetric matrix, A_2 a $(d-k) \times k$ matrix, $\mathbf{0}$ is the $(d-k) \times (d-k)$ zero matrix. Note that both A_1 and U are symmetric. It is clear that U has a dimension $d \times d$ (or $\alpha \times d$). An $[n, d]$ RS code is chosen to encode each row of U. The generator matrix of the RS code is given as

$$G = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ a^0 & a^1 & \cdots & a^{n-1} \\ (a^0)^2 & (a^1)^2 & \cdots & (a^{n-1})^2 \\ & & \vdots & \\ (a^0)^{k-1} & (a^1)^{k-1} & \cdots & (a^{n-1})^{k-1} \\ (a^0)^k & (a^1)^k & \cdots & (a^{n-1})^k \\ & & \vdots & \\ (a^0)^{d-1} & (a^1)^{d-1} & \cdots & (a^{n-1})^{d-1} \end{bmatrix}, \tag{9}$$

where a is a generator of $GF(2^m)$. Let C be the codeword vector with dimension $(\alpha \times n)$. It can be obtained as

$$U \cdot G = C.$$

From (9), G can be divided into two sub-matrices as

$$G = \begin{bmatrix} G_k \\ B \end{bmatrix}, \tag{10}$$

where

$$G_k = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ a^0 & a^1 & \cdots & a^{n-1} \\ (a^0)^2 & (a^1)^2 & \cdots & (a^{n-1})^2 \\ & & \vdots & \\ (a^0)^{k-1} & (a^1)^{k-1} & \cdots & (a^{n-1})^{k-1} \end{bmatrix} \tag{11}$$

and

$$B = \begin{bmatrix} (a^0)^k & (a^1)^k & \cdots & (a^{n-1})^k \\ & & \vdots & \\ (a^0)^{d-1} & (a^1)^{d-1} & \cdots & (a^{n-1})^{d-1} \end{bmatrix} .$$

It can be shown that G_k is a generator matrix of the $[n, k]$ RS code and it will be used in the decoding for data reconstruction.

III. Encoding and Decoding Schemes for MSR Codes

In this section, we propose a new encoding scheme for $[n, d]$ error-correcting MSR codes. With a feasible matrix Δ, $\bar{G}$ in (6) can be any generator matrix of the $[n, \alpha]$ RS code. The code construction in [14] is thus a special case of our proposed scheme. We can also select a suitable generator matrix such that the update complexity of the resulting code is minimized. An efficient decoding scheme is then proposed that uses the subcode of the $[n, d]$ RS code, the $[n, \alpha = k-1]$ RS code generated by $\bar{G}$, to perform the data reconstruction.

A. Encoding Schemes for Error-Correcting MSR Codes

RS codes are known to have very efficient decoding algorithms and exhibit good error correction capability. From (6) in Section II-B, a generator matrix G for MSR codes needs to satisfy:

1) $G = \begin{bmatrix} \bar{G} \\ \bar{G}\Delta \end{bmatrix}$, where $\bar{G}$ contains the first α rows in G and Δ is a diagonal matrix with distinct elements in the diagonal.
2) $\bar{G}$ is a generator matrix of the $[n, \alpha]$ RS code and G is a generator matrix of the $[n, d = 2\alpha]$ RS code.

Next, we present a sufficient condition for $\bar{G}$ and Δ such that G is a generator matrix of an $[n,d]$ RS code. We first introduce some notations. Let $g_{0y}(x)=\prod_{i=0}^{n-y-1}(x-a^i)$ and the $[n,y]$ RS code generated by $g_{0y}(x)$ be C_{0y}. Similarly, let $g_{1y}(x)=\prod_{i=1}^{n-y}(x-a^i)$ and the $[n,y]$ RS code generated by $g_{1y}(x)$ be C_{1y}. Clearly, $a^0,a^1,a^2,\ldots,a^{n-y-1}$ are roots of $g_{0y}(x)$, and $a^1,a^2,\ldots,a^{n-y}$ are roots of $g_{1y}(x)$. Thus, C_{0y} and C_{1y} are equivalent RS codes.

Theorem 1: Let $\bar{G}$ be a generator matrix of the $[n,\alpha]$ RS code $C_{0\alpha}$. Let the diagonal elements of Δ be $b_0,b_1,\ldots,b_{n-1}$ such that $b_i\neq b_j$ for all $i\neq j$, and $(b_0,b_1,\ldots,b_{n-1})$ is a codeword in $C_{1(\alpha+1)}$ but not $C_{1(\alpha)}$. In other words, $(b_0,b_1,\ldots,b_{n-1})\in C_{1(\alpha+1)}\backslash C_{1\alpha}$. Also let $\boldsymbol{c}\Delta\notin C_{0\alpha}$ for all nonzero $\boldsymbol{c}\in C_{0\alpha}$. Then, $G=\begin{bmatrix}\bar{G}\\ \bar{G}\Delta\end{bmatrix}$ is a generator matrix of the $[n,d]$ RS code C_{0d}.

Proof: We need to prove that each row of $\bar{G}\Delta$ is a codeword of C_{0d} and all rows in G are linearly independent. Let $\hat{C}_{0\alpha}$ be the dual code of $C_{0\alpha}$. It is well-known that $\hat{C}_{0\alpha}$ is an $[n,n-\alpha]$ RS code [17], [18]. Similarly, let $\hat{C}_{0d}$ be the dual code of C_{0d} and its generator matrix be H_d. Note that H_d is a parity-check matrix of C_{0d}. Let $h_d(x)=(x^n-1)/g_{0d}(x)$ and $h_\alpha(x)=(x^n-1)/g_{0\alpha}(x)$. Then, the roots of $h_d(x)$ and $h_\alpha(x)$ are $a^{n-d},a^{n-d+1},\ldots,a^{n-1}$ and $a^{n-\alpha},a^{n-\alpha+1},\ldots,a^{n-1}$, respectively. Since an RS code is also a cyclic code, then the generator polynomials of $\hat{C}_{0d}$ and $\hat{C}_{0\alpha}$ are $\hat{h}_d(x)$ and $\hat{h}_\alpha(x)$, respectively, where $\hat{h}_d(x)=x^{n-d}h_d(x^{-1})$ and $\hat{h}_\alpha(x)=x^{n-\alpha}h_\alpha(x^{-1})$. Clearly, the roots of $\hat{h}_d(x)$ are $a^{-(n-d)},a^{-(n-d+1)},\ldots,a^{-(n-1)}$ that are equivalent to $a^d,a^{d-1},\ldots,a^1$. Similarly, the roots of $\hat{h}_\alpha(x)$ are $a^\alpha,a^{\alpha-1},\ldots,a^1$. Since $\hat{h}_d(x)$ has roots of $a^d,a^{d-1},\ldots,a^1$, we can choose

$$H_d \;=\; \begin{bmatrix} 1 & 1 & \cdots & 1 \\ a^0 & a^1 & \cdots & a^{n-1} \\ (a^0)^2 & (a^1)^2 & \cdots & (a^{n-1})^2 \\ & & \vdots & \\ (a^0)^{n-d-1} & (a^1)^{n-d-1} & \cdots & (a^{n-1})^{n-d-1} \end{bmatrix} \qquad (12)$$

as the generator matrix of $\hat{C}_{0d}$. To prove that each row of $\bar{G}\Delta$ is a codeword of the RS code C_{0d} generated by G, it is sufficient to show that $\bar{G}\Delta H_d^T=\mathbf{0}$. From the symmetry of Δ, we have

$$\bar{G}\Delta H_d^T=\bar{G}\left(H_d\Delta\right)^T.$$

Thus, we only need to prove that each row of $H_d\Delta$ is a codeword in $\hat{C}_{0\alpha}$. Let the diagonal elements of Δ be $b_0,b_1,\ldots,b_{n-1}$. The ith row of $H_d\Delta$ is thus $r_i(x)=\sum_{j=0}^{n-1}b_j(a^j)^{i-1}x^j$ in the polynomial representation. Since each row of $H_d\Delta$ must be a codeword in $\hat{C}_{0\alpha}$, it follows that $r_i(a^\ell) = 0$ for $1 \le \ell \le \alpha$ and $1 \le i \le n-d$. Note that

$$r_i(a^\ell) = \sum_{j=0}^{n-1} b_j (a^j)^{i-1} (a^\ell)^j = \sum_{j=0}^{n-1} b_j (a^{i-1+\ell})^j \ .$$

Since $i-1+\ell$ runs from 1 to $n-d-1+\alpha = n-\alpha-1$, we have

$$\sum_{j=0}^{n-1} b_j (a^{\ell'})^j = 0 \text{ for } 1 \le \ell' \le n-\alpha-1 \ . \tag{13}$$

It is easy to see that the b_is satisfying (13) are all codewords in $C_{1(\alpha+1)}$.

The b_is need to make all rows in G linearly independent. Since all rows in $\bar{G}$ or those in $\bar{G}\Delta$ are linear independent, it is sufficient to prove that $C_{0\alpha} \cap C_\Delta = \{\mathbf{0}\}$, where C_Δ is the code generated by $\bar{G}\Delta$. Let $\boldsymbol{c}'$ be a codeword in C_Δ. $\boldsymbol{c}' = \boldsymbol{c}\Delta$ for some $\boldsymbol{c} \in C_{0\alpha}$. Consider some special codewords in $C_{0\alpha}$. It can be shown that, by the Mattson-Solomon polynomial [19], we can choose

$$\bar{G} = \begin{bmatrix} (a^0)^1 & (a^1)^1 & \cdots & (a^{n-1})^1 \\ (a^0)^2 & (a^1)^2 & \cdots & (a^{n-1})^2 \\ & & \vdots & \\ (a^0)^\alpha & (a^1)^\alpha & \cdots & (a^{n-1})^\alpha \end{bmatrix} \tag{14}$$

as the generator matrix of $C_{0\alpha}$. We can then select the ith row from $\bar{G}$ in (14) as $\boldsymbol{c}$, where $1 \le i \le \alpha$. Assume that $\boldsymbol{c}\Delta \in C_{0\alpha}$. Note that $\boldsymbol{c}\Delta(x)$ has $a^0, a^1, a^2, \ldots, a^{n-\alpha-1}$ as roots. Hence, we have

$$\sum_{j=0}^{n-1} b_j (a^{i+\ell})^j = 0 \text{ for } 0 \le \ell \le n-\alpha-1 \ . \tag{15}$$

From (13), it is easy to see that if $\sum_{j=0}^{n-1} b_j (a^{n-\alpha})^j = 0$, i.e., $a^{n-\alpha}$ is a root of $\sum_{j=0}^{n-1} b_j x^j$, then (15) holds for at least one i (e.g., $i = 1$). Thus, we need to exclude the codewords in $C_{1(\alpha+1)}$ that have $a^{n-\alpha}$ as root. These codewords turn out to be in $C_{1\alpha}$. ■

Corollary 1: Under the condition that the RS code is over $GF(2^m)$ for $m \ge \lceil \log_2 n \rceil$ and $\gcd(2^m-1, \alpha) = 1$, the diagonal elements of Δ, $b_0, b_1, \ldots, b_{n-1}$, can be

$$\gamma(a^0)^\alpha, \gamma(a)^\alpha, \gamma(a^2)^\alpha, \ldots, \gamma(a^{n-1})^\alpha \ ,$$

where $\gamma \in GF(2^m)\backslash\{\mathbf{0}\}$.

Proof: Note that one valid generator matrix of $C_{1(\alpha+1)}$ is

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ a^0 & a^1 & \cdots & a^{n-1} \\ (a^0)^2 & (a^1)^2 & \cdots & (a^{n-1})^2 \\ & & \vdots & \\ (a^0)^\alpha & (a^1)^\alpha & \cdots & (a^{n-1})^\alpha \end{bmatrix}. \tag{16}$$

$(b_0, b_1, \dots, b_{n-1}) \in C_{1(\alpha+1)} \backslash C_{1\alpha}$ can be represented as $b_i = \gamma(a^i)^\alpha + f_i$, where $(f_0, f_1, \dots, f_{n-1}) \in C_{1,\alpha}$. Now choose $(f_0, f_1, \dots, f_{n-1})$ to be all zero codeword. Under the condition that the RS code is over $GF(2^m)$ for $m \geq \lceil \log_2 n \rceil$ and $\gcd(2^m - 1, \alpha) = 1$, $\gamma(a^0)^\alpha, \gamma(a)^\alpha, \gamma(a^2)^\alpha, \dots, \gamma(a^{n-1})^\alpha$ is equivalent to $\gamma(a^\alpha)^0, \gamma(a^\alpha)^1, \gamma(a^\alpha)^2, \dots, \gamma(a^\alpha)^{n-1}$. If a^α is a generator of $GF(2^m)$, then all elements of $\gamma(a^\alpha)^0, \gamma(a^\alpha)^1, \gamma(a^\alpha)^2, \dots, \gamma(a^\alpha)^{n-1}$ are distinct. It is well-known that a^α is a generator if $\gcd(2^m - 1, \alpha) = 1$. Next we prove that $\boldsymbol{c}\Delta \notin C_{0\alpha}$ for all nonzero $\boldsymbol{c} \in C_{0\alpha}$ by contradiction. Let $\boldsymbol{c} = (c_0, c_1, \dots, c_{n-1})$. Then the polynomial representation of $\boldsymbol{c}\Delta$ is

$$\gamma \sum_{i=0}^{n-1} b_i c_i x^i = \gamma \sum_{i=0}^{n-1} c_i (a^\alpha x)^i \ .$$

Assume that $\boldsymbol{c}\Delta \in C_{0\alpha}$. Then $\gamma \sum_{i=0}^{n-1} c_i(a^\alpha x)^i$ must have roots $a^0, a^1, \dots, a^{n-\alpha-1}$. It follows that $c(x)$ must have $a^\alpha, a^{\alpha+1}, \dots, a^{n-1}$ as roots. Recall that $c(x)$ also has roots $a^0, a^1, a^2, \dots, a^{n-\alpha-1}$. Since $n - 1 \geq d = 2\alpha$, we have $n - \alpha - 1 \geq \alpha$. Hence, $c(x)$ has n roots of $a^0, a^1, a^2, \dots, a^{n-1}$. It is a contradiction since the degree of $c(x)$ is at most $n-1$. Thus, we conclude $\boldsymbol{c}\Delta \notin C_{0\alpha}$. ■

It is clear that by setting $\gamma = 1$ in Corollary 1, we obtain the generator matrix G given in (6) first proposed in [14] as a special case.[2]

One advantage of the proposed scheme is that it can now operate on a smaller finite field than that of the scheme in [14]. Another advantage is that one can choose $\bar{G}$ (and Δ accordingly) freely as long as $\bar{G}$ is the generator matrix of an $[n, \alpha]$ RS code. In particular, as discussed in Section I, to minimize the update complexity, it is desirable to choose a generator matrix that has the least row-wise maximum Hamming weight. Next, we present a least-update-complexity generator matrix that satisfies (6).

[2]Even though the roots in G given in (6) are different from those for the proposed generator matrix, they generate equivalent RS codes.

Corollary 2: Suppose Δ is chosen according to Corollary 1. Let $\bar{G}$ be the generator matrix associated with a systematic $[n, \alpha]$ RS code. That is,

$$\bar{G} = \begin{bmatrix} b_{00} & b_{01} & b_{02} & \cdots & b_{0(n-\alpha-1)} & 1 & 0 & 0 & \cdots & 0 \\ b_{10} & b_{11} & b_{12} & \cdots & b_{1(n-\alpha-1)} & 0 & 1 & 0 & \cdots & 0 \\ b_{20} & b_{21} & b_{22} & \cdots & b_{2(n-\alpha-1)} & 0 & & 1 & \cdots & 0 \\ & \vdots & & & & \vdots & & & & \vdots \\ b_{(\alpha-1)0} & b_{(\alpha-1)1} & b_{(\alpha-1)2} & \cdots & b_{(\alpha-1)(n-\alpha-1)} & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}, \tag{17}$$

where

$$x^{n-\alpha+i} = u_i(x)g(x) + b_i(x) \text{ for } 0 \le i \le \alpha - 1$$

and

$$b_i(x) = b_{i0} + b_{i1}x + \cdots + b_{i(n-\alpha-1)}x^{n-\alpha-1} .$$

Then, $G = \begin{bmatrix} \bar{G} \\ \bar{G}\Delta \end{bmatrix}$ is a least-update-complexity generator matrix.

Proof: The result holds since each row of $\bar{G}$ is a nonzero codeword with the minimum Hamming weight $n - \alpha + 1$. ■

B. *Efficient Decoding Scheme for MSR Codes*

Unlike the decoding scheme in [14] that uses $[n, d]$ RS code, we propose to use the subcode of the $[n, d]$ RS code, i.e., the $[n, \alpha = k-1]$ RS code generated by $\bar{G}$, to perform data reconstruction. The advantage of using the $[n, k-1]$ RS code is two-fold. First, its error correction capability is higher. Specifically, it can tolerate $\lfloor \frac{n-k+1}{2} \rfloor$ instead of $\lfloor \frac{n-d}{2} \rfloor$ errors. Second, it only requires the access of two additional storage nodes (as opposed to $d - k + 2 = k$ nodes) for each extra error.

Without loss of generality, we assume that the data collector retrieves encoded symbols from $k + 2v$ ($v \ge 0$) storage nodes, $j_0, j_1, \dots, j_{k+2v-1}$. We also assume that there are v storage nodes whose received symbols are erroneous. The stored information on the $k + 2v$ storage nodes are collected as the $k + 2v$ columns in $Y_{\alpha \times (k+2v)}$. The $k + 2v$ columns of G corresponding to storage nodes $j_0, j_1, \dots, j_{k+2v-1}$ are denoted as the columns of G_{k+2v}. First, we discuss data reconstruction when $v = 0$. The decoding procedure is similar to that in [13].

No Error: In this case, $v = 0$ and there is no error in Y. Then,

$$\begin{aligned} Y_{\alpha\times k} &= UG_k \\ &= [Z_1 Z_2] \begin{bmatrix} \bar{G}_k \\ \bar{G}_k \Delta \end{bmatrix} \\ &= \left[Z_1 \bar{G}_k + Z_2 \bar{G}_k \Delta\right] . \end{aligned} \tag{18}$$

Multiplying $\bar{G}_k^T$ to both sides of (18), we have [13],

$$\begin{aligned} \bar{G}_k^T Y_{\alpha\times k} &= \bar{G}_k^T U G_k \\ &= [\bar{G}_k^T Z_1 \bar{G}_k + \bar{G}_k^T Z_2 \bar{G}_k \Delta] \\ &= P + Q\Delta . \end{aligned} \tag{19}$$

Since Z_1 and Z_2 are symmetric, P and Q are symmetric as well. The (i, j)th element of $P + Q\Delta$, $1 \le i, j \le k$ and $i \ne j$, is

$$p_{ij} + q_{ij} a^{(j-1)\alpha} \ , \tag{20}$$

and the (j, i)th element is given by

$$p_{ji} + q_{ji} a^{(i-1)\alpha} \ . \tag{21}$$

Since $a^{(j-1)\alpha} \ne a^{(i-1)\alpha}$ for all $i \ne j$, $p_{ij} = p_{ji}$, and $q_{ij} = q_{ji}$, combining (20) and (21), the values of p_{ij} and q_{ij} can be obtained. Note that we only obtain $k - 1$ values for each row of P and Q since no elements in the diagonal of P or Q are obtained.

To decode P, recall that $P = \bar{G}_k^T Z_1 \bar{G}_k$. P can be treated as a portion of the codeword vector, $\bar{G}_k^T Z_1 \bar{G}$. By the construction of $\bar{G}$, it is easy to see that $\bar{G}$ is a generator matrix of the $[n, k-1]$ RS code. Hence, each row in the matrix $\bar{G}_k^T Z_1 \bar{G}$ is a codeword. Since we know $k - 1$ components in each row of P, it is possible to decode $\bar{G}_k^T Z_1 \bar{G}$ by the error-and-erasure decoder of the $[n, k-1]$ RS code.[3]

Since one cannot locate any erroneous position from the decoded rows of P, the decoded α codewords are accepted as $\bar{G}_k^T Z_1 \bar{G}$. By collecting the last α columns of $\bar{G}$ as $\bar{G}_\alpha$ to find its

[3] The error-and-erasure decoder of an $[n, k-1]$ RS code can successfully decode a received vector if $s + 2v < n - k + 2$, where s is the number of erasure (no symbol) positions, v is the number of errors in the received portion of the received vector, and $n - k + 2$ is the minimum Hamming distance of the $[n, k-1]$ RS code.

inverse (here it is an identity matrix), one can recover $\bar{G}_k^T Z_1$ from $\bar{G}_k^T Z_1 \bar{G}$. Since any α rows in $\bar{G}_k^T$ are independent and thus invertible, we can pick any α of them to recover Z_1. Z_2 can be obtained similarly by Q.

Single Error: In this case, $v = 1$ and only one column of $Y_{\alpha\times(k+2)}$ is erroneous. Without loss of generality, we assume the erroneous column is the first column in Y. That is, the symbols received from storage node j_0 contain error. Let $E = \left[\boldsymbol{e}_1^T|\mathbf{0}\right]$ be the error matrix, where $\boldsymbol{e}_1 = [e_{11}, e_{12,}, \ldots, e_{1\alpha}]$ and $\mathbf{0}$ is all zero matrix with dimension $\alpha \times (k+1)$. Then

$$\begin{aligned} Y_{\alpha\times(k+2)} &= UG_{k+2} + E \\ &= [Z_1 Z_2] \begin{bmatrix} \bar{G}_{k+2} \\ \bar{G}_{k+2}\Delta \end{bmatrix} + E \\ &= [Z_1\bar{G}_{k+2} + Z_2\bar{G}_{k+2}\Delta] + E \ . \end{aligned} \tag{22}$$

Multiplying $\bar{G}_{k+2}^T$ to both sides of (22), we have

$$\begin{aligned} \bar{G}_{k+2}^T Y_{\alpha\times(k+2)} &= \bar{G}_{k+2}^T U G_{k+2} + \bar{G}_{k+2}^T E \\ &= [\bar{G}_{k+2}^T Z_1 \bar{G}_{k+2} + \bar{G}_{k+2}^T Z_2 \bar{G}_{k+2}\Delta] + \bar{G}_{k+2}^T E \\ &= P + Q\Delta + \left[\bar{G}_{k+2}^T \boldsymbol{e}_1^T|\mathbf{0}\right] \\ &= \tilde{P} + \tilde{Q}\Delta \ . \end{aligned} \tag{23}$$

It is easy to see that the errors only affect the first column of $\tilde{P} + \tilde{Q}\Delta$ since the nonzero elements are all in the first column of $\left[\bar{G}_{k+2}^T \boldsymbol{e}_1^T|\mathbf{0}\right]$. Similar to (20) and (21), the values of $\tilde{p}_{ij}$ and $\tilde{q}_{ij}$, where $i \neq j$, are obtained from $\bar{G}_{k+2}^T Y_{\alpha\times(k+2)}$ even though there are some errors in them. Note that we only obtain $k+1$ values for each row of $\tilde{P}$ and $\tilde{Q}$. Since the $(j,1)$th elements of $\bar{G}_{k+2}^T Y_{\alpha\times(k+2)}$ may be erroneous for $1 \leq j \leq k+2$, the values calculated from them contain errors as well. Then the first column and the first row of $\tilde{P}$ ($\tilde{Q}$) have errors. Note that each row of $\tilde{P}$ ($\tilde{Q}$) has only at most one error except the first row.

First, we decode $\tilde{P}$. Recall that $P = \bar{G}_{k+2}^T Z_1 \bar{G}_{k+2}$. As mentioned earlier, P can be treated as a portion of the codeword vector $\bar{G}_{k+2}^T Z_1 \bar{G}$, and then $\tilde{P}$ can be decoded by the $[n, k-1]$ RS code. Since we have obtained $k+1$ components in each row of $\tilde{P}$, it is possible to correctly decode each row of $\bar{G}_{k+2}^T Z_1 \bar{G}$, except for the first row of $\tilde{P}$, using the error-and-erasure decoder of the RS code.

Let $\hat{P}$ be the corresponding portion of decoded codeword vector to $\tilde{P}$ and $E_P = \hat{P} \oplus \tilde{P}$ be the error pattern vector. Next we describe how to locate the incorrect row after decoding every row (in this case we assume that the error occurs in the first row). Now suppose that there are more than two errors in the first column of $\tilde{P}$.[4] Let these errors be in $(j_1, 1)$th, $(j_2, 1)$th,$\cdots$, and $(j_\ell, 1)$th positions in $\tilde{P}$. After decoding all rows of $\tilde{P}$, it is easy to see that all rows but the first row can be decoded correctly due to at most one error occurring in each row. Then one can confirm that the number of nonzero elements in E_P in the first column is at least three since only the error in the first position of the first column can be decoded incorrectly. Other than the first column in E_P there is at most one nonzero element in rest of the columns. Then the first column in $\hat{P}$ has correct elements except the one in the first row. Just copy all elements in the first column of $\hat{P}$ to those corresponding positions of its first row to make $\hat{P}$ a symmetric matrix. We then collect any α columns of $\hat{P}$ except the first column as $\hat{P}_\alpha$ and find its corresponding $\bar{G}_\alpha$. By multiplying the inverse of $\bar{G}_\alpha$ to $\hat{P}_\alpha$, one can recover $\bar{G}^T_{k+2} Z_1$. Since any α rows in $\bar{G}^T_{k+2}$ are independent and thus invertible, we can pick any α of them to recover Z_1. Z_2 can be obtained similarly by Q.

Multiple Errors: Before presenting the proposed decoding algorithm, we first prove that a decoding procedure can always successfully decode Z_1 and Z_2 if $v \leq \lfloor \frac{n-k+1}{2} \rfloor$ and all storage nodes are accessed. Assume the storage nodes with errors correspond to the ℓ_0th, ℓ_1th, $\ldots$, ℓ_{v-1}th columns in the received matrix $Y_{\alpha\times n}$. Then,

$$\begin{aligned}
& \bar{G}^T Y_{\alpha\times n} \\
= \; & \bar{G}^T U G + \bar{G}^T E \\
= \; & \bar{G}^T [Z_1 Z_2] \begin{bmatrix} \bar{G} \\ \bar{G}\Delta \end{bmatrix} + \bar{G}^T E \\
= \; & [\bar{G}^T Z_1 \bar{G} + \bar{G}^T Z_2 \bar{G}\Delta] + \bar{G}^T E \ ,
\end{aligned} \tag{24}$$

where

$$E = \left[\mathbf{0}_{\alpha\times(\ell_0-1)} | \boldsymbol{e}^T_{\ell_0} | \mathbf{0}_{\alpha\times(\ell_1-\ell_0-1)} | \cdots | \boldsymbol{e}^T_{\ell_{v-1}} | \mathbf{0}_{\alpha\times(n-\ell_{v-1})} \right] \ .$$

[4] It will be shown later that the number of errors in the first column of $\tilde{P}$ is at least three.

Lemma 1: There are at least $n-k+2$ errors in each of the ℓ_0th, ℓ_1th, $\ldots$, ℓ_{v-1}th columns of $\bar{G}^T Y_{\alpha\times n}$.

Proof: From (24), we have

$$\bar{G}^T Y_{\alpha\times n} = P + Q\Delta + \bar{G}^T E.$$

The error vector in ℓ_jth column is then

$$\bar{G}^T \boldsymbol{e}_{\ell_j}^T = \left(\boldsymbol{e}_{\ell_j}\bar{G}\right)^T \quad . \tag{25}$$

Since $\bar{G}$ is a generator matrix of the $[n,k-1]$ RS code, $\boldsymbol{e}_{\ell_j}\bar{G}$ in (25) is a nonzero codeword in the RS code. Hence, the number of nonzero symbols in $\boldsymbol{e}_{\ell_j}\bar{G}$ is at least $n-k+2$, the minimum Hamming distance of the RS code. ■

We next have the main theorem to perform data reconstruction.

Theorem 2: Let $\bar{G}^T Y_{\alpha\times n} = \tilde{P} + \tilde{Q}\Delta$. Furthermore, let $\hat{P}$ be the corresponding portion of decoded codeword vector to $\tilde{P}$ and $E_P = \hat{P} \oplus \tilde{P}$ be the error pattern vector. Assume that the data collector accesses all storage nodes and there are v, $1 \le v \le \lfloor \frac{n-k+1}{2} \rfloor$, of them with errors. Then, there are at least $n-k+2-v$ nonzero elements in ℓ_jth column of E_P, $0 \le j \le v-1$, and at most v nonzero elements in the rest of the columns of E_P.

Proof: Let us focus on the ℓ_jth column of E_P. By Lemma 1, there are at least $n-k+2$ errors in the ℓ_jth column of $\bar{G}^T Y_{\alpha\times n}$. $\tilde{P}$ is constructed from $\bar{G}^T Y_{\alpha\times n}$ based on (20) and (21). If there is only one value of (20) and (21) that is in error, then the constructed p_{ij} and q_{ij} will be in error. However, when both values are in error, p_{ij} and q_{ij} might accidentally be correct. Among those $n-k+2$ erroneous positions, there are at least $n-k+2-v$ positions in error after constructing $\tilde{P}$ since at most v errors can be corrected in constructing $\tilde{P}$. It is easy to see that at least $n-k+2-v$ positions are in error that are not among any of the ℓ_0th, ℓ_1th, $\ldots$, ℓ_{v-1}th elements in the ℓ_jth column. These errors are in rows that can be decoded correctly. Hence, there are at least $n-k+2-v$ errors that can be located in ℓ_jth column of $\tilde{P}$ such that there are at least $n-k+2-v$ nonzero elements in the ℓ_jth column of E_P. There are at most v rows in $\tilde{P}$ that cannot be decode correctly due to having more than v errors in each of them. Hence, other than those columns with errors in the original matrix $\bar{G}^T Y_{\alpha\times n}$, at most v errors will be found in each of the rest of the columns of $\tilde{P}$. ■

The above theorem allows us to design a decoding algorithm that can correct up to $\lfloor\frac{n-k+1}{2}\rfloor$ errors.[5] In particular, we need to examine the erroneous positions in $\bar{G}^T E$. Since $1 \le v \le \lfloor\frac{n-k+1}{2}\rfloor$, we have $n-k+2-v \ge \lfloor\frac{n-k+1}{2}\rfloor + 1 > v$. Thus, the way to locate all erroneous columns in $\tilde{P}$ is to find out all columns in E_P where the number of nonzero elements in them are greater than or equal to $\lfloor\frac{n-k+1}{2}\rfloor + 1$. After we locate all erroneous columns we can follow a procedure similar to that given in the no error (or single error) case to recover Z_1 from $\hat{P}$.

The above decoding procedure guarantees to recover Z_1 (Z_2) when all n storage nodes are accessed. However, it is not very efficient in terms of bandwidth usage. Next, we present a progressive decoding version of the proposed algorithm that only accesses enough extra nodes when necessary. Before presenting it, we need the following corollary.

Corollary 3: Consider that one accesses $k+2v$ storage nodes, among which v nodes are erroneous and $1 \le v \le \lfloor\frac{n-k+1}{2}\rfloor$. There are at least $v+2$ nonzero elements in the ℓ_Jth column of E_P, $0 \le j \le v-1$, and at most v among the remaining columns of E_P.

Proof: This is a direct result from Theorem 2 when we delete $n-(k+2v)$ elements in each column of E_P according to the size of $Y_{\alpha\times(k+2v)}$ and $n-k+2-v-\{n-(k+2v)\} = v+2$. ■

Based on Corollary 3, we can design a progressive decoding algorithm [20] that retrieves extra data from the remaining storage nodes when necessary. To handle Byzantine fault tolerance, it is necessary to perform integrity check after the original data is reconstructed. Two verification mechanisms have been suggested in [14]: cyclic redundancy check (CRC) and cryptographic hash function. Both mechanisms introduce redundancy to the original data before they are encoded and are suitable to be used in combination with the decoding algorithm.

The progressive decoding algorithm starts by accessing k storage nodes. Error-and-erasure decoding succeeds only when there is no error. If the integrity check passes, then the data collector recovers the original data. If the decoding procedure fails or the integrity check fails, then the data collector retrieves two more blocks of data from the remaining storage nodes. Since the data collector has $k+2$ blocks of data, the error-and-erasure decoding can correctly recover the original data if there is only one erroneous storage node among the $k+1$ nodes accessed. If the integrity check passes, then the data collector recovers the original data. If the decoding

[5] In constructing $\tilde{P}$ we only get $n-1$ values (excluding the diagonal). Since the minimum Hamming distance of an $[n, k-1]$ RS code is $n-k+2$, the error-and-erasure decoding can only correct up to $\lfloor\frac{n-1-k+2}{2}\rfloor$ errors.

procedure fails or the integrity check fails, then the data collector retrieves two more blocks of data from the remaining storage nodes. The data collector repeats the same procedure until it recovers the original data or runs out of the storage nodes. The detailed decoding procedure is summarized in Algorithm 1.

IV. Efficient Encoding and Decoding schemes for MBR Codes

In this section, we will find a generator matrix of the form (10) such that the row with the maximum Hamming weight has the least number of nonzero elements. This generator matrix is thus a least-update-complexity matrix. A decoding scheme for MBR codes that can correct more error patterns is also provided.

A. *Encoding Scheme for MBR Codes*

Let $g(x) = \prod_{j=1}^{n-k}(x - a^j) = \sum_{i=0}^{n-k} g_i x^i$ be the generator polynomial of the $[n, k]$ RS code and $f(x) = \prod_{j=1}^{n-d}(x - a^j) = \sum_{i=0}^{n-d} f_i x^i$ the generator polynomial of the $[n, d]$ RS code, where a is a generator of $GF(2^m)$. A matrix G can be constructed as

$$G = \begin{bmatrix} G_k \\ B \end{bmatrix}, \tag{26}$$

where

$$G_k = \begin{bmatrix} g_0 & g_1 & \cdots & g_{n-k} & 0 & 0 & \cdots & 0 \\ 0 & g_0 & \cdots & g_{n-k-1} & g_{n-k} & 0 & \cdots & 0 \\ & & & \vdots & & & & \\ 0 & \cdots & 0 & g_0 & g_1 & g_2 & \cdots & g_{n-k} \end{bmatrix} \tag{27}$$

and

$$B = \begin{bmatrix} f_0 & f_1 & \cdots & f_{n-d} & 0 & 0 & \cdots & 0 & 0 \\ 0 & f_0 & \cdots & f_{n-d-1} & f_{n-d} & 0 & \cdots & 0 & 0 \\ & & & \vdots & & & & & \\ 0 & \cdots & 0 & f_0 & \cdots & f_{n-d} & 0 & \cdots & 0 \end{bmatrix}. \tag{28}$$

The dimensions of G_k and B are $k \times n$ and $(d - k) \times n$, respectively. Next, we prove that the main theorem about the rank of G given in (26).

Theorem 3: The rank of G given in (26) is d. That is, it is a generator matrix of the MBR code.

Proof: Let the codes generated by G_k and G be $\bar{C}$ and C, respectively. It can be seen that any row in G_k and B is a cyclic shift of the previous row. Hence, all rows in G_k and B are linearly independent. Now we only consider the linear combination of rows in G chosen from both G_k and B. Since $\bar{C}$ is a linear code, the portion of the linear combination that contains only rows from G_k results in a codeword, named $\boldsymbol{c}$, in $\bar{C}$. Assume that the rows chosen from B are the j_0th, j_1th, $\ldots$, and $j_{\ell-1}$th rows. Recall that B can be represented by a polynomial matrix as

$$B(x) = \begin{bmatrix} f(x) \\ xf(x) \\ x^2 f(x) \\ \vdots \\ x^{d-k-1} f(x) \end{bmatrix} .$$

Hence, in the polynomial form, the linear combination can be represented as

$$\boldsymbol{c}(x) + \sum_{i=0}^{\ell-1} b_i x^{j_i - 1} f(x) \ , \tag{29}$$

where $\boldsymbol{c}(x)$ is not the all-zero codeword and not all $b_i = 0$. Since $c(x)$ is the code polynomial of $\bar{C}$, it is divisible by $g(x)$ and can be represented as $u(x)g(x)$. Assume that (29) is zero. Then we have

$$u(x)g(x) = -f(x) \sum_{i=0}^{\ell-1} b_i x^{j_i - 1} \ . \tag{30}$$

Recall that $g(x) = \prod_{i=1}^{n-k}(x - a^i)$ and $f(x) = \prod_{i=1}^{n-d}(x - a^i)$. Hence,

$$g(x) = f(x) \prod_{i=n-d+1}^{n-k} (x - a^i) \ . \tag{31}$$

Substituting (31) into (30) we have

$$u(x) \prod_{i=n-d+1}^{n-k} (x - a^i) = -\sum_{i=0}^{\ell-1} b_i x^{j_i - 1} \ . \tag{32}$$

That is, $\sum_{i=0}^{\ell-1} b_i x^{j_i-1}$ is divisible by $\prod_{i=n-d+1}^{n-k}(x - a^i)$. However, the degree of $\prod_{i=n-d+1}^{n-k}(x-a^i)$ is $d-k$ and the degree of $\sum_{i=0}^{\ell-1} b_i x^{j_i-1}$ is at most $d-k-2$ when $\ell = d-k-1$, the largest possible value for ℓ. Thus, $\sum_{i=0}^{\ell-1} b_i x^{j_i-1}$ is not divisible by $\prod_{i=n-d+1}^{n-k}(x-a^i)$ since not all $b_i = 0$. This is a contradiction.

Since all rows in G_k and B are codewords in C, G is then a generator matrix of the $[n, d]$ RS code C. ■

Corollary 4: The G given in (26) is the least-update-complexity matrix.

Proof: Since G_k must be the generator matrix of the $[n, k]$ RS code $\bar{C}$, the Hamming weight of each row of G_k is greater than or equal to the minimum Hamming distance of $\bar{C}$, $n-k+1$. Since the degree of $g(x)$ is $n-k$ and itself is a codeword in $\bar{C}$, the nonzero coefficients of $g(x)$ is $n-k+1$ and each row of G_k is with $n-k+1$ Hamming weight. A similar argument can be applied to each row of B such that the Hamming weight of it is $n-d+1$. Thus, the G given in (26) has the least number of nonzero elements. Further, Since G_k is the generator matrix of the $[n, k]$ code, the minimum Hamming of its row can have is $n-k+1$, namely, the minimum Hamming distance of the code. Hence, the row with maximum Hamming weight in G is $n-k+1$. ■

Since $\bar{C}$ is also a cyclic code, it can be arranged as a systematic code. G_k is then given by

$$\boldsymbol{G} = \begin{bmatrix} b_{00} & b_{01} & b_{02} & \cdots & b_{0(n-k-1)} & 1 & 0 & 0 & \cdots & 0 \\ b_{10} & b_{11} & b_{12} & \cdots & b_{1(n-k-1)} & 0 & 1 & 0 & \cdots & 0 \\ b_{20} & b_{21} & b_{22} & \cdots & b_{2(n-k-1)} & 0 & & 1 & \cdots & 0 \\ & \vdots & & & & \vdots & & & & \vdots \\ b_{(k-1)0} & b_{(k-1)1} & b_{(k-1)2} & \cdots & b_{(k-1)(n-k-1)} & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}, \qquad (33)$$

where

$$x^{n-k+i} = u_i(x)g(x) + b_i(x) \text{ for } 0 \le i \le k-1,$$

and $b_i(x) = b_{i0} + b_{i1}x + \cdots + b_{i(n-k-1)}x^{n-k-1}$. It is easy to see that G with G_k as a submatrix is still a least-update-complexity matrix. The advantage of a systemic code will become clear in the decoding procedure of the MBR code.

B. Decoding Scheme for MBR Codes

The generator polynomial of the RS code encoded by (33) has $a^{n-k}, a^{n-k-1}, \ldots, a$ as roots. Hence, the progressive decoding scheme based on the $[n, k]$ RS code given in [14] can be applied to decode the MBR code. The decoding algorithm given in [14] is slightly modified as follows.

Assume that the data collector retrieves encoded symbols from ℓ storage nodes $j_0,\ j_1, \ldots,\ j_{\ell-1}$, $k \leq \ell \leq n$. The data collector receives d vectors where each vector has ℓ symbols. Denoting the first k vectors among the d vectors as $Y_{k\times\ell}$ and the remaining $d-k$ vectors as $Y_{(d-k)\times\ell}$. By the encoding of the MBR code, the codewords in the last $d-k$ rows of C can be viewed as encoded by G_k instead of G. Hence, the decoder of the $[n,k]$ RS code can be applied on $Y_{(d-k)\times\ell}$ to recover the codewords in the last $d-k$ rows of C.

Let $\tilde{C}_{(d-k)\times k}$ be the last k columns of the codewords recovered by the error-and-erasure decoder in the last $d-k$ rows of C. Since the code generated by (33) is a systematic code, A_2 in U can be reconstructed as

$$\tilde{A}_2 = \tilde{C}_{(d-k)\times k}\ . \tag{34}$$

We then calculate the j_0th, j_1th, ..., $j_{\ell-1}$th columns of $\tilde{A}_2^T \cdot B$ as $E_{k\times\ell}$, and subtract $E_{k\times\ell}$ from $Y_{k\times\ell}$:

$$Y'_{k\times\ell} = Y_{k\times\ell} - E_{k\times\ell}\ . \tag{35}$$

Applying the error-and-erasure decoding algorithm of the $[n,k]$ RS code again on $Y'_{k\times\ell}$ we can reconstruct A_1 as

$$\tilde{A}_1 = \tilde{C}_{k\times k}\ . \tag{36}$$

The decoded information sequence is then verified by data integrity check. If the integrity check is passed, the data reconstruction is successful; otherwise the progressive decoding procedure is applied, where two more storage nodes need to be accessed from the remaining storage nodes in each round until no further errors are detected.

The decoding capability of the above decoding algorithm is $\lfloor\frac{n-k+1}{2}\rfloor$. Since each erroneous storage node sends $\alpha = d$ symbols to the data collector, in general, not all α symbols are wrong if failures in the storage nodes are caused by random faults. Hence, the decoding algorithm given in [14] can be modified as follows to extend error correction capability. After decoding $Y_{(d-k)\times\ell}$, one can locate the erroneous columns of $Y_{(d-k)\times\ell}$ by comparing the decoded result to it. Assume that there are v erroneous columns located. Delete the corresponding columns in $E_{k\times\ell}$ and $Y_{k\times\ell}$ and we have

$$Y'_{k\times(\ell-v)} = Y_{k\times(\ell-v)} - E_{k\times(\ell-v)}\ . \tag{37}$$

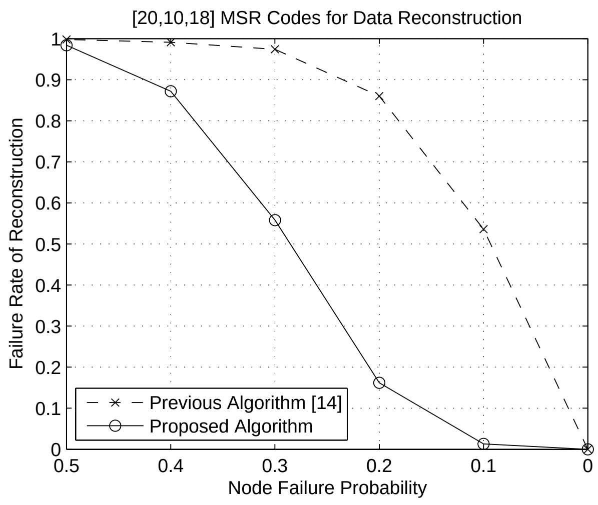

Fig. 1. Comparison of the failure rate between the algorithm in [14] and the proposed algorithm for $[20, 10, 18]$ MSR codes

Applying the error-and-erasure decoding algorithm of the $[n, k]$ RS code again on $Y'_{k\times(\ell-v)}$ to reconstruct A_1 if $\ell-v \geq k$; otherwise the progressive decoding is applied. The modified decoding algorithm is summarized in Algorithm 2. The advantage of the modified decoding algorithm is that it can correct errors up to

$$\left\lfloor \frac{n-k+1}{2} \right\rfloor + \left\lfloor \frac{n-k+1-\lfloor\frac{n-k+1}{2}\rfloor}{2} \right\rfloor$$

even though not all error patterns up to such number of errors can be corrected.

V. Performance Evaluation

The proposed reconstruction algorithms for MSR and MBR codes have been evaluated by Monte Carlo simulations. They are compared with the reconstruction algorithms previously proposed in [14]. Each data point is generated from 10^3 simulation runs. Storage nodes may fail arbitrarily with the Byzantine failure probability ranging from 0 to 0.5. In both schemes, $[n, k, d]$ and m are chosen to be $[20, 10, 18]$ and 5, respectively.

In the first set of simulations, we compare the proposed algorithm with the algorithm in [14] in terms of the failure rate of the reconstruction and the average number of node accesses, which indicates the required bandwidth for data reconstruction. Failure rate is defined as the percentage of runs for which reconstruction fails (due to insufficient number of healthy storage

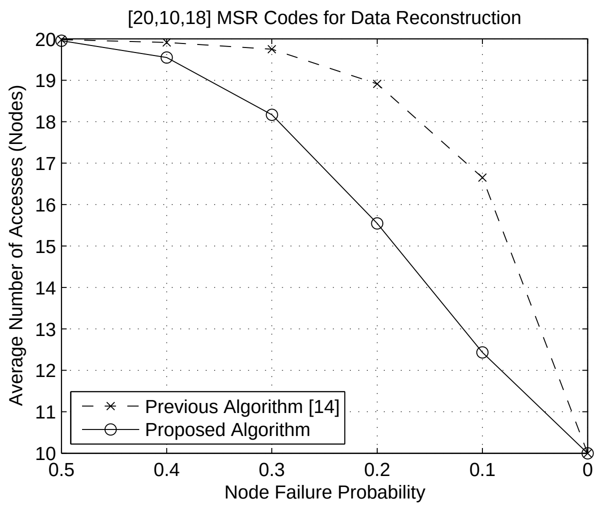

Fig. 2. Comparison of the number of node accesses between the algorithm in [14] and the proposed algorithm for $[20, 10, 18]$ MSR codes

nodes). Figure 1 shows that the proposed algorithm can successfully reconstruct the data with much higher probability than the previous algorithm for the same node failure probability. For example, when the node failure probability is 0.1, only about 1% of the time, reconstruction fails using the proposed algorithm, in contrast to 50% with the old algorithm. The advantage of the proposed algorithm is also pronounced in the average number of accessed nodes for data reconstruction, as illustrated in Fig. 2. For example, on an average, only 2.5 extra nodes are needed by the proposed algorithm under the node failure probability of 0.1; while over 6.5 extra nodes are required by the old algorithm in [14]. It should be noted that the actual saving attained by the new algorithm depends on the setting of n, k, d and the number of errors.

The previous and proposed decoding algorithms for MBR codes are compared in the second set of simulations. Figures 3 and 4 show that they have identical failure rates of reconstruction and average number of accessed nodes. This result implies that the special error patterns, which only the proposed algorithm is able to handle for successful data reconstruction, do not happen very frequently. However, the computational complexity of the proposed algorithm for MBR encoding is much lower since no matrix inversion and multiplications are needed in (34) and (36).

In the evaluation of the update complexity, we adopt the measure given in [16]. The update

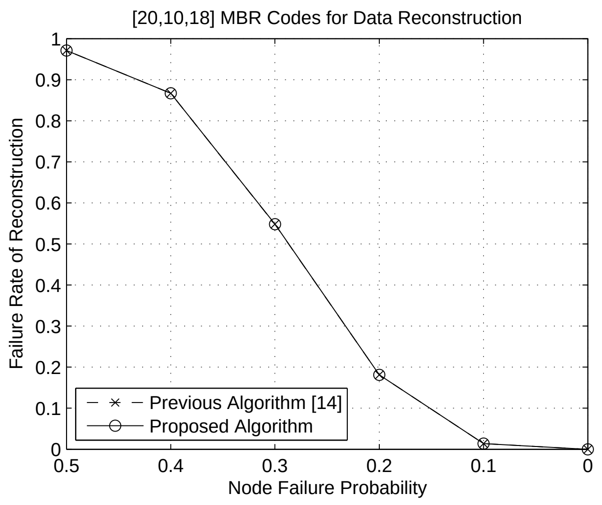

Fig. 3. Failure-rate comparison between the previous algorithm in [14] and the proposed algorithm for $[20, 10, 18]$ MBR codes

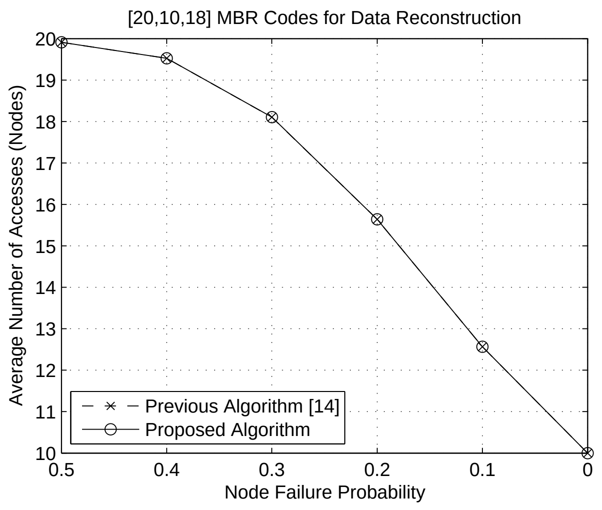

Fig. 4. Node-access comparison between the previous algorithm in [14] and the proposed algorithm for $[20, 10, 18]$ MBR codes

complexity is measured by the maximum number of nonzero elements in all rows of the generator matrix G. Denote by $\eta(R)$ the ratio of the update complexity of the proposed generator matrix to that of the generator matrix given in [14], where $R = k/n$. It is easy to see that,

$$\eta_{MSR}(R) = \frac{n - \alpha + 1}{n} \approx 1 - R$$

for MSR codes and

$$\eta_{MBR}(R) = \frac{n-k+1}{n} \approx 1 - R$$

for MBR codes. Thus, the proposed encoding schemes incur only $1-R$ of the update complexity of that in [14].

VI. Conclusion

In this work, we proposed new encoding and decoding schemes for the $[n, d]$ error-correcting MSR and MBR codes that generalize the previously proposed codes in [14]. Through both theoretical analysis and numerical simulations, we demonstrated the superb error correction capability, low update complexity and low computation complexity of the new codes.

It is easy to see that there is a trade-off between the update complexity and error correction capability of regenerating codes. In this work, we fist find good error-correcting regenerating codes and then optimize their update complexity. Possible future work includes the study of encoding schemes that first design regenerating codes with good update complexity and then optimize their error correction capability.

References

[1] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," in *Proc. of the 19th ACM SIGOPS Symp. on Operating Systems Principles*, Bolton Landing, NY, October 2003.

[2] J. K. et al., "OceanStore: an architecture for global-scale persistent storage," in *Proc. of the 9th International Conference on Architectural Support for programming Languages and Operating Systems*, Cambridge, MA, November 2000.

[3] R. Bhagwan, K. Tati, Y. Cheng, S. Savage, and G. Voelker, "Total recall: system support for automated availability management," in *Proc. of the 1st Conf. on Networked Systems Design and Implementation*, San Francisco, CA, March 2004.

[4] A. G. Dimakis, P. B. Godfrey, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," in *Proc. of 26th IEEE International Conference on Computer Communications (INFOCOM)*, Anchorage, Alaska, May 2007, pp. 2000–2008.

[5] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inform. Theory*, vol. 56, pp. 4539 – 4551, September 2010.

[6] Y. Wu, A. G. Dimakis, and K. Ramchandran, "Deterministic regenerating codes for distributed storage," in *Proc. of 45th Annual Allerton Conference on Control, Computing, and Communication*, Urbana-Champaign, Illinois, September 2007.

[7] Y. Wu, "Existence and construction of capacity-achieving network codes for distributed storage," *IEEE Journal on Selected Areas in Communications*, vol. 28, pp. 277 – 288, February 2010.

[8] D. F. Cullina, "Searching for minimum storage regenerating codes," California Institute of Technology Senior Thesis, 2009.

[9] Y. Wu and A. G. Dimakis, "Reducing repair traffic for erasure coding-based storage via interference alignment," in *Proc. IEEE International Symposium on Information Theory*, Seoul, Korea, July 2009, pp. 2276–2280.

[10] K. V. Rashmi, N. B. Shah, P. V. Kumar, and K. Ramchandran, "Explicit construction of optimal exact regenerating codes for distributed storage," in *Proc. of 47th Annual Allerton Conference on Control, Computing, and Communication*, Urbana-Champaign, Illinois, September 2009, pp. 1243–1249.

[11] S. Pawar, S. E. Rouayheb, and K. Ramchandran, "Securing dynamic distributed storage systems against eavesdropping and adversarial attacks," arXiv:1009.2556v2 [cs.IT] 27 Apr 2011.

[12] F. Oggier and A. Datta, "Byzantine fault tolerance of regenerating codes," arXiv:1106.2275v1 [cs.DC] 12 Jun 2011.

[13] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction," *IEEE Trans. Inform. Theory*, vol. 57, pp. 5227–5239, August 2011.

[14] Y. S. Han, R. Zheng, and W. H. Mow, "Exact regenerating codes for byzantine fault tolerance in distributed storage," in *Proc. of the IEEE INFOCOM 2012*, Orlendo, FL, March 2012.

[15] K. Rashmi, N. Shah, K. Ramchandran, and P. Kumar, "Regenerating codes for errors and erasures in distributed storage," in *Proc. of the 2012 IEEE International Symposium on Information Theory*, Cambridge, MA, July 2012.

[16] A. S. Rawat, S. Vishwanath, A. Bhowmick, and E. Soljanin, "Update efficient codes for distributed storage," in *Proc. of the 2011 IEEE International Symposium on Information Theory*, Saint Petersburg, Russia, July 2011.

[17] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, Inc., 2004.

[18] T. K. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*. Hoboken, NJ: John Wiley & Sons, Inc., 2005.

[19] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. New York, NY: Elsevier Science Publishing Company, Inc., 1977.

[20] Y. S. Han, S. Omiwade, and R. Zheng, "Progressive data retrieval for distributed networked storage," *IEEE Trans. on Parallel and Distributed Systems*, vol. 23, pp. 2303–2314, December 2012.

Algorithm 1: Decoding of MSR Codes Based on $(n, k-1)$ RS Code for Data Reconstruction

```
begin
    $v = 0$; $j = k$;
    The data collector randomly chooses $k$ storage nodes and retrieves encoded data, $Y_{\alpha\times j}$;
    while $v \leq \lfloor \frac{n-k+1}{2} \rfloor$ do
        Collect the $j$ columns of $\bar{G}$ corresponding to accessed storage nodes as $\bar{G}_j$;
        Calculate $\bar{G}_j^T Y_{\alpha\times j}$;
        Construct $\tilde{P}$ and $\tilde{Q}$ by using (20) and (21);
        Perform progressive error-and-erasure decoding on each row in $\tilde{P}$ to obtain $\hat{P}$;
        Locate erroneous columns in $\hat{P}$ by searching for columns of them with at least
        $v+2$ errors; assume that $\ell_e$ columns found in the previous action;
        Locate columns in $\hat{P}$ with at most $v$ errors; assume that $\ell_c$ columns found in the
        previous action;
        if ($\ell_e = v$ and $\ell_c = k+v$) then
            Copy the $\ell_e$ erronous columns of $\hat{P}$ to their corresponding rows to make $\hat{P}$ a
            symmetric matrix;
            Collect any $\alpha$ columns in the above $\ell_c$ columns of $\hat{P}$ as $\hat{P}_\alpha$ and find its
            corresponding $\bar{G}_\alpha$;
            Multiply the inverse of $\bar{G}_\alpha$ to $\hat{P}_\alpha$ to recover $\bar{G}_j^T Z_1$;
            Recover $Z_1$ by the inverse of any $\alpha$ rows of $\bar{G}_j^T$;
            Recover $Z_2$ from $\tilde{Q}$ by the same procedure; Recover $\tilde{\boldsymbol{m}}$ from $Z_1$ and $Z_2$;
            if integrity-check($\tilde{\boldsymbol{m}}$) = SUCCESS then
                return $\tilde{\boldsymbol{m}}$;
        $j \leftarrow j+2$;
        Retrieve $2$ more encoded data from remaining storage nodes and merge them into
        $Y_{\alpha\times j}$; $v \leftarrow v+1$;
    return FAIL;
```

Algorithm 2: Decoding of MBR Codes for Data Reconstruction

```
begin
    The data collector randomly chooses $k$ storage nodes and retrieves encoded data, $Y_{d\times k}$;
    $\ell \leftarrow k$;
    repeat
        Perform progressive error-erasure decoding on last $d-k$ rows in $Y_{d\times\ell}$, $Y_{(d-k)\times\ell}$, to
        recover $\tilde{C}$ (error-erasure decoding performs $d-k$ times);
        Locate the erroneous columns in $Y_{(d-k)\times\ell}$ (assume to have $v$ columns);
        Calculate $\tilde{A}_2$ via (34);
        Calculate $\tilde{A}_2 \cdot B$ and obtain $Y'_{k\times(\ell-v)}$ via (37);
        if ($\ell - v \geq k$) then
            Perform progressive error-erasure decoding on $Y'_{k\times(\ell-v)}$ to recover the first $k$
            rows in codeword vector (error-erasure decoding performs $k$ times);
            Calculate $\tilde{A}_1$ via (36);
            Recover the information sequence $\boldsymbol{m}$ from $\tilde{A}_1$ and $\tilde{A}_2$;
            if integrity-check($\boldsymbol{m}$) = SUCCESS then
                return $\boldsymbol{m}$;
        $\ell \leftarrow \ell + 2$;
        Retrieve two more encoded data from remaining storage nodes and merge them
        into $Y_{d\times\ell}$;
    until $\ell \geq n-2$;
    return FAIL;
```